

# Calibration Management

- Top Level
  - DB structure
  - Manager code
  - Subdetector issues
- Updating
  - DB structure
  - Program
  - Miscellaneous
- Status

# Top Level: DB Structure

- **D0\_OFFLINE\_CALIBRATION\_SETS**
  - D0 and subdetector calibration
  - Ids for each valid combination
  - Subdet ID's are NOT foreign keys (yet)
- **D0\_MY\_DBIDS**
  - Used to satisfy bookkeeping via EnvID
  - One row/database
  - Ids defined on calib/align web page
- **D0\_OFFLINE\_CALIBRATIONS**
  - Defines validity ranges in terms of run and lumi (and version)
    - Rectangles (run,lumi) space
  - Foreign keys into previous tables

# Top Level: Manager Code

- Provides
  - Steering (with manual overrides)
  - I/O specification and initialization
- Three classes
  - **Calibration\_management** (and its rcp)  
trivial framework package  
knows about event, but work done in
  - **CalibrationManager**  
not a framework package  
singleton  
all work (except Event) in here  
called by calibration\_management  
uses db\_server\_gen'd classes for tables  
calls subdetector xRecoCalibrators with ID
  - **CalibDetector**  
performs translation between RecoXCalibrator  
Ids and ids actually found in DB

# Top Level: Subdetector Issues

- Subdets accessed via xRecoCalibrater
  - No Physical coupling (easy insertion)
  - Found from CalibrationRegistry using ID passed as xRecoCalibrater template
  - xRecoCalibrater template ID must be ID for production DB

<http://d0server1.projects.OfflineCalibAlign/home.html>

- How is a subdetector registered in CalibrationRegistry?
  - In (bottom of) xRecoCalibration.cpp include a line like (e.g. for SMT)  
`SmtRecoCalibrater* register_me = SmtRecoCalibrater::get_instance();`
  - Access a RecoCalibrater method somewhere in your reco code

# Updating: DB Structure

- **D0\_CALIBRATION\_UPDATES**
  - Rows added by subdet yellow boxes
    - Insert subdet id, run and lumi range and calib id (for subdet)
    - There is a sequence define: NEWUPD to be used for unique ID for inserts
  - Periodic scan/process/delete program
  - How to handle access? Roles, accounts
- **D0\_CALIBRATION\_UPDATE\_LOGS**
  - Summary messages from program
- **D0\_CALIBRATION\_OLD\_UPDATES**
  - Historical copy of update table

# Updating: Program

- External program
  - Reads all updates from table
  - Combines overlapping (run,lumi) regions from different subdets
  - Builds new d0 calibration sets and generates d0 calibration ids
  - Build new “validity range” records
  - Inserts sets and ranges into master tables
  - Removes processed updates (copy into record table)
- Python script, update\_calibrations in calibration\_management.

# Updating: Miscellaneous

- Certain conventions must be followed for run and lumi
- Propose the following:
  - Runs
    - If upper bound unknown, set to  
**MAX RUN = 20000000**
  - Luminosity (?)
    - Python seems to want integers(?)
    - Specify in units of 10E29 (e.g. 10e31 = 100)
    - with  
**MAX LUMI = 20000000**
    - Currently no Dzero standard units and not in event record**

# Status

- Tables generated and in dev
- Server tested using python and C++
- Reco management software debugged and in CVS
- Updating program 90% complete (maybe an afternoon's work remains. Spent 1.5 days so far)
- Report generation incomplete
- Comments?